IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 09/595,003	§	Examiner:	Pillai, Namitha
Filed: June 13, 2000	§	Group/Art Unit:	2173
Inventor(s):	§	Atty. Dkt. No:	5150-44300
NICOLAS VAZQUEZ,	§		
JEFFREY L. KODOSKY,	§		
RAM KUDUKOLI,	§		
KEVIN L. SCHULTZ,	§		
DINESH NAIR, and	§		
CHRISTOPHE CALTAGIRONE	§		
Title: SYSTEM AND METHOD	§		
FOR AUTOMATICALLY	§		
GENERATING A	§		
GRAPHICAL PROGRAM	§		
TO IMPLEMENT A	§		
PROTOTYPE			

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed May 22, 2007, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504, as evidenced by the assignment recorded at Reel 014052, Frame 0095.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this Appeal.

III. STATUS OF CLAIMS

Claims 1-7, 9-37, 39-59, 61-74, and 76-90 are pending. Claims 8, 38, 60, and 75 are cancelled. Claims 1-7, 9-37, 39-59, 61-74, and 76-90 are rejected and are the subject of this Appeal Brief. A copy of claims 1-7, 9-37, 39-59, 61-74, and 76-90 as on appeal is included in the Claims Appendix attached hereto.

IV. STATUS OF AMENDMEMNTS

No amendments to the claims have been filed subsequent to the rejection in the Office Action of March 22, 2007. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a method of automatically creating a graphical program to perform an algorithm. A graphical program is a software program whose source code is not textual in nature, but rather the source code is graphical in nature, e.g., a plurality of nodes or icons connected by wires.

One or more functions or operations of an algorithm or process may be recorded in response to user input, e.g., selecting or performing the one or more functions or operations of the algorithm or process. *See, e.g.,* Specification p. 24, lines 12-26; Figure 6 at 304. A graphical program which implements the one or more functions or operations of the algorithm or process may then be automatically generated in response to or based on the recorded functions. *See, e.g.,* Specification p. 24, lines 27-30; Figure 6 at 306. In automatically generating the graphical program, a plurality of interconnected nodes which visually indicate functionality of the graphical program may be automatically included in the graphical program without direct user input selecting the nodes. *See, e.g.,* Specification p. 24, line 30 - p. 25, line 3; p. 71, lines 12-14. In other words, the nodes are automatically included in the graphical program in an automatic or programmatic manner, without any direct user input required to select the nodes or place the nodes in the graphical program.

Independent claim 31 is directed to a system for automatically creating a graphical program to perform an algorithm. The system may include a processor or CPU. *See, e.g.,* Specification p. 20, line 10; Figure 3 at 160. The system may include a user input device. *See, e.g.,* Specification p. 76, line 3. The system may include a memory coupled to the processor which stores a prototyping environment application. *See, e.g.,* Specification p. 20, lines 13-16; Figure 3 at 166, 164, and 162. The prototyping environment application may be executable to perform the method of claim 1, the subject matter of which is summarized above.

Independent claim 53 is directed to a memory medium comprising program instructions executable to perform the method of claim 1, the subject matter of which is summarized above.

Independent claim 71 is directed to a method of creating a graphical program to perform an algorithm. The method includes creating a prototype in response to user input where the prototype specifies the algorithm. *See, e.g.,* Specification p. 24, lines 5-14; Figure 6 at 302 and 304. The method also includes automatically generating the

graphical program in response to the prototype. *See, e.g.,* Specification p. 24, lines 27-30; Figure 6 at 306. The graphical program implements the algorithm. *See, e.g.,* Specification p. 26, lines 1-3. The graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program. *See, e.g.,* Specification p. 21, lines 19-21. In automatically generating the graphical program, the nodes are automatically included in the graphical program, i.e., are placed in the graphical program automatically. Hence, no direct user input is required to place the nodes in the graphical program, as they are automatically included. Also, no direct user input is required to select the nodes when they are being automatically included in the graphical program. *See, e.g.,* Specification p. 24, line 30 - p. 25, line 3; p. 71, lines 12-14.

Independent claim 81 is directed to a memory medium comprising program instructions executable to perform the method of claim 71, the subject matter of which is summarized above.

Independent claim 90 is directed to a memory medium comprising program instructions executable to record one or more functions in response to user input, where the one or more functions specify an algorithm. *See, e.g.,* Specification p. 24, lines 12-26; Figure 6 at 304. The memory medium further comprises program instructions executable to automatically generate a graphical program in response to the recorded one or more functions. *See, e.g.,* Specification p. 24, lines 27-30; Figure 6 at 306. The graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, and the graphical program implements the algorithm. *See, e.g.,* Specification p. 21, lines 19-21; Specification p. 26, lines 1-3. In automatically generating the graphical program, the program instructions are further executable to automatically generate graphical code in the graphical program without direct user input. *See, e.g.,* Specification p. 24, line 30 - p. 25, line 3; p. 71, lines 12-14. In other words, the user is not required to manually specify the graphical code, but rather the graphical code is automatically generated by the program instructions.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-7, 9-16, 21-37, 39-43, 45-59, 61-65, 67-74, and 76-90 are finally rejected under 35 U.S.C. 103(a) as being unpatentable over Morris et al. (U.S. Patent No. 5,862,372, hereinafter "Morris"), Oka et al. (EP Publication No. 0510514 A1, "Oka"), and Yamada (US Patent No. 4,831,580).

2. Claims 17-20, 44, and 66 are finally rejected under 35 U.S.C. 103(a) as being unpatentable over Morris, Oka, and Yamada, in further view of Shi et al. (U.S. Patent No. 5,623,659, hereinafter "Shi").

VII. ARGUMENT

First Ground of Rejection:

Claims 1-7, 9-16, 21-37, 39-43, 45-59, 61-65, 67-74, and 76-90 are finally rejected under 35 U.S.C. 103(a) as being unpatentable over Morris, Oka, and Yamada. Appellant respectfully traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 5, 7, 10, 13, 42, 53, 57, 59, 62, 64, and 90

Appellant respectfully submits that each of the independent claims 1, 53, and 90 recites one or more features not taught or suggested in Morris, Oka, and Yamada, taken singly or in combination.

More specifically, the cited art does not teach or suggest: "automatically generating the graphical program in response to the recorded one or more functions, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm, wherein said automatically generating the graphical program comprises automatically including the nodes in the graphical program, wherein said

automatically including the nodes in the graphical program is performed without direct user input selecting the nodes (*emphasis added*)" as recited in claim 1.

The Examiner relied on Morris as teaching various elements of the claims. However, Morris neither teaches nor suggests automatic creation of a graphical program as recited in the present claims. Rather, Morris teaches that a user of the system can manually develop an application. Morris states that: "Development of a complete application is accomplished by visually arranging, ordering, and interconnecting the objects without the necessity of writing any code" (Morris Abstract). The reference in Morris to "without the necessity of writing any code" refers to without the necessity of textual coding, i.e., without the necessity of writing textual code. Morris does teach (and only teaches) manual creation of graphical code, NOT automatic creation. More specifically, Morris teaches "the user of the system drags an object into the view. The system simultaneously generates the underlying script which reflects the object and the properties associated with the object" (Morris col. 6, lines 23-26) (*emphasis added*).

Appellant respectfully notes that, as admitted in the previous Office Action, Morris fails to teach or suggest automatically generating a graphical program, including automatically including nodes in the graphical program without direct user input selecting the nodes. More specifically, Morris teaches "the user of the system drags an object into the view. The system simultaneously generates the underlying script which reflects the object and the properties associated with the object" (Morris col. 6, lines 23-26) (*emphasis added*). The Office Action specifically states, "Morris does not disclose that user input does not use the selection of the nodes. [sic]"

The Office Action asserts that Oka and Yamada remedy this admitted deficiency of Morris. More specifically, the Office Action asserts that Oka discloses "automatically generating a graphical program represented as the flowchart with the program functions, where the blocks or nodes representing each function is automatically generated, selected and formed into a graphical program without any user intervention with the graphical program being automatically generated", citing col.1:20-26, which reads:

> It is an object of the present invention to **automatically draw a flow chart as a graphic representation of data processing contents** in accordance with definition information defining the data processing

> contents, thereby allowing easy understanding of the data processing contents. (*emphasis added*)

As the cited text makes clear, and as explained in the previous Response, Oka is directed to automatically generating a *flowchart* that graphically represents data processing contents, *not* a graphical program (which, being a program, is by definition executable to perform some specified functionality). Applicant submits that automatic generation of a graphical program, which is an executable program, is far more complex and quite different than merely generating a picture (a flowchart) as taught in Oka. Oka further clarifies this aspect of Oka's invention in col.1:47-col.2:12, which describes the system for automatically generating the flowchart:

> e) forming means for, in response to a start of flow chart formation, forming processing block patterns by adding the names of the units of processing to the basic block pattern in the respective units of processing stored in said processing sequence storage means; and (f) output means for sequentially connecting the processing block patterns, formed in accordance with the respective units of processing stored in said processing sequence storage means, in accordance with the sequence stored in said processing sequence storage means, and **outputting the resultant block patterns on a recording paper sheet**.
>
> According to the present invention, since **a flow chart graphically representing a processing outline can be automatically drawn, there is no need to output a program list or manually draw a flow chart** as in the prior art. That is, a flow chart which can be understood by anybody can be very easily obtained. Especially, if sets of graphic patterns are formed or processing flow charts are graphically drawn in units of processing, the resultant **illustration** is easier to understand. (*emphasis added*)

Appellant respectfully submits that the Office Action has mischaracterized Oka, and notes that the product of Oka's method is not a graphical program, but rather, a flowchart, specifically, a printout of a flowchart that *illustrates* the specified data processing process. As the cited text describes, Oka's system and method automatically generate the flowchart using connected block patterns that represent or illustrate block units of processing, which are themselves (the block units of processing) stored in processing sequence storage means. These block patterns are simply standard graphical flowchart shapes used to represent various steps or processes, specifically the block units

of processing, e.g., as shown in Figure 22 of Oka, and are specifically *not* graphical program nodes. The automatically generated flowchart is nowhere described as being a program. Moreover, in the Response to Arguments, the Office Action admits that Oka does not disclose "generating and creating a program", i.e., that Oka's flowchart is not a program.

In direct contrast, Appellant respectfully notes that per Appellant's disclosure (e.g., p.33, line 7, and elsewhere) and dependent claims 10 and 62, Appellant's automatically generated graphical program is *executable*. Moreover, as noted above, programs are executable, while flowcharts are illustrative, and are *not* executable, as is known to those of skill in the art of programming.

The Office Action admits that Morris and Oka do not disclose that the automatically created flowchart is generated based on an algorithm, where the flowchart generation involves creation of a program, but asserts that Yamada remedies this admitted deficiency by disclosing a flowchart which when created, generates a program, citing col.2:8-27, which reads:

> The above object can be accomplished by providing a program generator, according to the present invention, comprising:
> means for displaying at least symbols for flow-chart elements;
> means for inputting a programming language;
> first means of converting a language inputted by said language input means into flow-chart element symbols;
> flow-chart editing means by which a flow-chart can be depicted on said display means and edited as desired through operation of said language input means; and
> second means for converting into a predetermined programming language a program corresponding to said edited flow-chart;
> wherein the flow-chart displayed on said display means is corrected and applied with additions or deletions until it is made complete; then a program is generated in a predetermined language corresponding to the complete flow-chart.

Appellant respectfully submits that while Yamada does disclose generating a program from a flowchart, the programs of Yamada are nowhere described as graphical programs, and in fact, are clearly text-based programs. For example, col.10:30-35 reads thusly:

> ...the program generator is used herein with a language system for a programmable controller, but C language or BASIC language, namely a conventional language system, may be used instead. Of course, any appropriate simple language system may be used.

Thus, Yamada clearly indicates that the automatically generated programs may be in a programmable controller language, or a conventional language system, or "any appropriate simple language system". Appellant respectfully submits that Appellant's graphical program is not a program in a conventional language system, i.e., a text-based language, such as C or BASIC. Nor is Appellant's graphical program a program in a "simple language system". As is well-known in the art of programming, graphical programming languages, such as graphical data flow languages, are significantly different from conventional or simple programming languages, at least for the reason that graphical programs are spatial (2D) in nature, as opposed to the linear nature of text-based programs. Appellant respectfully notes that the "graphic program" and "graphic program code" described in Yamada refers to a program or program code that executes to display graphics, specifically, a flowchart on a CRT, and does not refer to a graphical program as claimed. Thus, Yamada fails to disclose automatically generating a graphical program.

Additionally, in the Response to Arguments, the Office Action states that "the flowchart of Oka is based on a predetermined program, where the flowchart is generated based on the program", and that "Yamada has disclosed a flowchart, which is generated to create the program, where this flowchart upon creation automatically generates a program". Appellant agrees with the Examiner that the flowchart of Oka is based on a predetermined program (or programs), as described in Oka, col.1:1-25:

> In a data processing apparatus such as an office computer, work **processing operation programs** corresponding to the work contents of a user are generally designed in advance. When the **processing contents** are to be checked later, a program list is output or a flow chart graphically representing the processing outline is manually drawn.
>
> A program list, however, is too technical and hence difficult for general users to understand. On the other hand, drawing of a flow chart takes a lot

of time and labor even for a system engineer, demanding a very laborious task.

It is an object of the present invention to automatically **draw a flow chart as a graphic representation of data processing contents** in accordance with definition information defining the data processing contents, thereby allowing easy understanding of the data processing contents. (*emphasis added*)

Similarly, col.4:58-col.5 reads:

…the flow chart forming device 13 fetches and **analyzes the contents of various programs prestored in the PS processing device 11, reads out graphic block patterns from the block pattern memory 12 in accordance with the analysis result**, and sets the patterns in a development buffer 15. Note that the development buffer 15 is capable of storing, e.g., data corresponding to one A4-size sheet. The data in the development buffer 15 is supplied to the PS processing device 11 and is output from a printing unit in the PS processing 11. **With this operation, a flow chart graphically representing a processing outline is formed**… (*emphasis added*)

Col.5:32-34 states:

…**processing contents are defined as programs** in various units of processing…

As the above text makes clear, the processing programs are analyzed to generate the flowchart, and so necessarily pre-exist the flowchart. Note that per Oka, the generated flowchart is provided in place of a program listing, e.g., for ease of understanding the program. Thus, Oka's system operates to generate a flowchart based on a predetermined program (or programs), as the Office Action states. Appellant thus respectfully submits that these programs are necessary for Oka's technique to generate the flowchart, and thus Oka teaches away from Appellant's invention as represented in claim 1, since according to claim 1, the graphical program is automatically generated based on a one or more functions recorded in response to user input, where the one or more functions specify an algorithm. Note that according to claim 1, the recorded functions are not nodes in the graphical program, i.e., the user input does not select the nodes.

Thus, Appellant respectfully submits that according to the alleged combination of Morris, Oka, and Yamada, the user manually creates a graphical program (Morris), then a program is created and analyzed to automatically generate a flowchart (Oka), then a flowchart is used to automatically generate a conventional program (Yamada). Clearly, this alleged combination does not, and cannot, teach Appellant's invention as represented in claim 1. None of the cited art teaches or suggests, or even hints at, automatically generating a graphical program as claimed.

Appellant further submits that the cited art fails to teach or suggest "recording one or more functions in response to user input, wherein the one or more functions specify the algorithm", where the functions are distinct from the graphical program nodes, as claimed. For example, in Morris, the user manually selects the graphical icons or nodes and drags them onto the diagram or program, in Oka, the flowchart is generated based on a pre-existing program, not user selected functions, and in Yamada, the (conventional) program is generated based on a user specified flowchart. Thus, the cited art fails to disclose automatically generating a graphical program based on recorded functions specified by user input.

Appellant also respectfully submits that there is no teaching, suggestion, or motivation to combine Morris, Oka, and Yamada in the references. As held by the U.S. Court of Appeals for the Federal Circuit in *Ecolochem Inc. v. Southern California Edison Co.*, an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis. Furthermore, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular. . .Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination.

Appellant respectfully submits that the motivation to combine Oka and Morris suggested by the Office Action with respect to claim 1 appears to simply be an assertion

that both Oka and Morris are directed to generating graphical programs, and that Oka discloses doing so automatically, which, as explained above, is incorrect, since Oka discloses automatic generation of a flowchart, not a graphical program, and Morris discloses manually generating a graphical program. Thus, the suggested motivation to combine is improper, and Oka and Morris are not available in combination to make a prima facie case of obviousness.

Appellant respectfully submits that the motivation to combine suggested by the Office Action in the Response to Arguments: "automatic generation of such a structure representing generation of a program can be convenient" (from Oka), is not only directed to automatic generation of a flowchart, and thus not germane to Appellant's invention, but is not a clear and particular suggestion, teaching, or motivation, to combine with Morris and Yamada with respect to the features and limitations of claim 1. Nowhere does Oka or Morris suggest the desirability of automatically generating a graphical program. Nor does Yamada indicate the desirability, or even possibility, of automatically generating a graphical program.

Further, even were Morris, Oka, and Yamada, properly combinable, which Appellant argues they are not, the resulting combination would not produce the features of claim 1. Thus, Appellant respectfully submits that a case of *prima facie* obviousness has not been established to reject claim 1. Accordingly, Appellant respectfully submits that, at least for the reasons presented, claim 1 is patentably distinct and nonobvious over the cited art, and is thus allowable.

Claims 53 and 90 each includes limitations similar to claim 1, specifically, the feature ". . .wherein said automatically generating the graphical program comprises automatically generating graphical code in the graphical program without direct user input", and so the arguments presented above apply with equal force to these claims, as well. Accordingly, Appellant respectfully submits that, at least for the reasons presented, claims 53 and 90 are patentably distinct and nonobvious, and thus allowable.

Claims 2 and 54

Appellant respectfully submits that neither Morris nor Oka teaches or suggests ". . .performing the one or more functions in response to user input. . ." and

". . .wherein said recording the one or more functions is performed in response to said performing the one or more functions" as recited by claim 2.

The Examiner stated in the Office Action mailed August 26, 2004: ". . .Morris discloses performing the function in response to user input, wherein the user dragging the objects is the function which is then recorded in response to the user's action for specifying the algorithm (column 3, lines 32-34)" (*emphasis added*).

Appellant respectfully submits that dragging an object is user input, not a function to be performed in response to user input. The "functions" in this claim refer to the functions required to perform the algorithm, and which are implemented in the generated program. For example, the user input may invoke functions that may include one or more image processing functions on an image, in response to which the image processing functions are performed, and the one or more image processing functions recorded. Thus, ". . .the user dragging the objects. . ." is not ". . .performing the one or more functions in response to user input. . ." as recited by claim 2.

In the Response to Arguments, the Examiner asserts that a "function can be a set of steps taken to carry out a process, thereby the user dragging and dropping is interpreted as a processing function carried out in association with the program. [sic]". Appellant respectfully submits that the Examiner has improperly attempted to redefine "function" in contradiction with the clear meaning of both the claims and the Specification. One of skill in the art would readily understand that the user dragging and dropping is not a processing function. Additionally, Applicant submits that there would be no point in recording "dragging and dropping" instead of processing functions, since it is the functions that specify the algorithm, i.e., the dragging and dropping is simply the means whereby the user selects or indicates the desired functionality of the generated graphical program. Nowhere does the Specification indicate that it is the dragging and dropping that is recorded.

Thus, Morris fails to teach or suggest this feature of claims 2 and 54, and so Appellant respectfully submits that claims 2 and 54 are patentably distinct and non-obvious over Morris, and thus allowable.

Claims 9, 61

Appellant respectfully submits that neither Morris nor Oka teaches or suggests ". . .wherein said automatically generating the graphical program comprises automatically including and connecting the nodes in the graphical program without direct user input (*emphasis added*)."

In contrast, in Appellant's invention as recited in claim 9, the program icons or nodes are automatically included in the program and connected together without any direct user input selecting the nodes or connecting them together. Neither Morris nor Oka teaches or suggests this feature. For example, as discussed above, Morris discloses manually creating a graphical program, and Oka discusses automatically generating a flowchart, which is not a graphical program, and thus, the cited art fails to teach or suggest this feature.

Thus, Appellant respectfully submits that claim 9 is patentably distinguished over both Morris and Oka, taken both singly and in combination. Accordingly, Appellant respectfully submits that, at least for the reasons presented, claim 9 is allowable.

Claim 61 includes limitations similar to claim 9, and so the arguments presented above apply with equal force to this claim, as well. Appellant respectfully submits that, for at least the reasons presented above, claim 61 is patentably distinguished over both Morris and Oka, taken both singly and in combination, and is thus allowable.

Claims 11, 40, and 63

Appellant respectfully submits that Morris nowhere teaches or suggests "wherein the graphical program includes a block diagram portion and a user interface panel portion", as recited in claim 11.

The Examiner relied upon "the palette in [Morris'] Figure 5" to teach "wherein the graphical program includes. . .a user interface panel portion" of Appellant's claim 11.

Appellant respectfully submits that the palette in Morris' Figure 5 is a part of an integrated development environment (IDE), which receives user input during manual creation of a script. More particularly, the IDE shown in Morris' Figure 5 is not part of a graphical program which comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program. Rather Morris teaches:

> FIG. 5 shows for a simple program all four views, Output, Map, Multitrack, and Workform simultaneously displayed in separate windows. In each view, the system of this invention permits development of the application using the same method of adding an object, interconnecting the object, and setting the objects properties. For instance, from a palette of objects, which displays a collection of icons that represent each object available to the system (such as the palette shown in FIG. 6), the user of the system drags an object into the view. The system simultaneously generates the underlying script which reflects the object and the properties associated with the object. (Morris col. 6, lines 15-26) (*emphasis added*)

Furthermore, Appellant respectfully submits that Morris neither teaches nor suggests that the script created, which is not a graphical program, includes a block diagram portion and a user interface panel portion. Nor do Oka or Yamada disclose this feature.

In the Response to Arguments, the Examiner asserts that the various windows used to present views of the diagram and the palette from which the user selects icons teach the claims user interface panel portion of a graphical program. Appellant respectfully disagrees. As explained previously and above, the cited windows and palette are GUI elements of the development environment, and are not part of the graphical program itself, and so Morris fails to teach or suggest this feature of claim 11.

Thus, for at least this reason, Appellant respectfully submits that claim 11 is patentably distinguished over the cited art, and is thus allowable.

Claims 40 and 63 include limitations similar to claim 11, and so the arguments presented above apply with equal force to these claims, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 40, and 63 are patentably distinguished over the cited art, and are thus allowable.

Claim 12

In addition to the features and limitations of claim 1, discussed above, claim 12 includes the limitation "wherein the graphical program is a graphical data flow program."

Nowhere do any of the cited references even mention a graphical data flow program. Thus, for at least these reasons, Appellant respectfully submits that claim 12 is patentably distinguished over the cited art, and thus allowable.

Claims 14, 43, and 65

Appellant respectfully submits that Morris nowhere teaches or suggests the combinations of features ". . .modifying the script to create a new script in response to user input after said creating the association. . ." and ". . .modifying the graphical program according to the new script to create a new graphical program" as recited in claim 14.

Rather, Morris' Abstract discloses: "The system generates as output a script listing the objects and their properties which is then executed by a separate run time program" (*emphasis added*). Furthermore, Morris also teaches and discloses that "For instance, from a palette of objects, which displays a collection of icons that represent each object available to the system (such as the palette shown in FIG. 6), the user of the system drags an object into the view. The system simultaneously generates the underlying script which reflects the object and the properties associated with the object" (Morris col. 6, lines 20-26) (*emphasis added*).

In other words, Morris teaches and discloses that the user drags objects into the view and then, in response and according to dragging the objects into the view, a script is simultaneously generated. In contrast, Appellant's invention as recited in claim 14 includes ". . .modifying the script to create a new script in response to user input after said creating the association. . ." and ". . .modifying the graphical program according to the new script to create a new graphical program". Thus, Morris teaches a direction divergent from the path that was taken by Appellant. Therefore, Morris is seen as teaching away from Appellant's present claims. Accordingly, Appellant respectfully submits that a *prima facie* case of obviousness has not been established to reject claim 14. *See In re Haruna*, 249 F.3d 1327, 1335 (Fed. Cir. 2001) (outlining elements of a reference teaching away and rebutting a *prima facie* case of obviousness). Thus, Morris fails to disclose this feature, as do Oka and Yamada.

Accordingly, Appellant respectfully submits that, at least for the reasons presented, claim 14 is patentably distinct and non-obvious over the cited art, and thus allowable.

Claims 43 and 65 each includes limitations similar to claim 14, and so the arguments presented above apply with equal force to these claims, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 43 and 65 are patentably distinguished over the cited art, and are thus allowable.

Claim 15

Appellant respectfully submits that Morris neither teaches nor suggests "wherein said modifying the graphical program according to the new script uses the association between the script and the graphical program; wherein the association remains between the new script and the new graphical program" as recited in claim 15

Rather, Morris teaches and discloses "For instance, from a palette of objects, which displays a collection of icons that represent each object available to the system (such as the palette shown in FIG. 6), the user of the system drags an object into the view. The system simultaneously generates the underlying script which reflects the object and the properties associated with the object" (Morris col. 6, lines 20-26) (*emphasis added*).

Nowhere does Morris (or Oka or Yamada) disclose ". . .wherein said modifying the graphical program according to the new script uses the association between the script and the graphical program", nor "wherein the association remains between the new script and the new graphical program", as claimed (*emphasis added*). Thus, Appellant respectfully submits that claim 15 is patentably distinguished over the cited art, and is thus allowable.

Claim 16

Appellant respectfully submits that Morris (nor Oka or Yamada) neither teaches nor suggests "receiving user input indicating a desire to change the graphical program; displaying script information of the script; modifying the script information in response to user input; and modifying the graphical program after said modifying the script information.

Appellant respectfully submits that Morris discloses: "First, the system can be used to author applications using an entirely visual programming scheme (paradigm) which does not require the user to know or be able to write any specialized code. Icons representing the objects (and accordingly their functionalities) may be placed (dragged) into an appropriate view" (Morris col. 3, lines 30-33) (*emphasis added*).

In other words, Morris teaches and discloses using "an entirely visual programming scheme" while Appellant invention recites ". . .modifying the script information in response to user input. . ." and ". . .modifying the graphical program after said modifying the script information" as recited in claim 16. ". Thus, Morris teaches a direction divergent from the path that was taken by Appellant. Therefore, Morris is seen as teaching away from Appellant's present claims. Accordingly, Appellant respectfully submits that a *prima facie* case of obviousness has not been established to reject claim 16. *See In re Haruna*, 249 F.3d 1327, 1335 (Fed. Cir. 2001) (outlining elements of a reference teaching away and rebutting a *prima facie* case of obviousness). In particular, Morris does not teach or suggest that user modification of a script can cause a corresponding graphical program to be automatically modified accordingly.

Appellant respectfully submits that, at least for the above reasons presented, claim 16 is allowable.

Claims 21, 45, and 67

Appellant respectfully submits that Morris (nor Oka nor Yamada) neither teaches nor suggests "receiving user input specifying code generation information; wherein said automatically generating the graphical program utilizes the code generation information", as recited in claim 21.

Morris nowhere discloses allowing a user to specify code generation information this is then used to affect an automatic graphical program generation process. As noted above, Morris does not teach or suggest any type of automatic generation of a graphical program (as admitted in the Office Action), and certainly does not allow a user to specify code generation information to affect an automatic graphical program generation process. Thus, Appellant respectfully submits that claim 21 is patentably distinct and non-obvious over the cited art, and is thus allowable.

Claims 45 and 67 each includes limitations similar to claim 21, and so the arguments presented above apply with equal force to these claims, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 45 and 67 are patentably distinguished over the cited art, and are thus allowable.

Claims 22, 23, 46, 47, and 68

Appellant respectfully submits that Morris nowhere teaches or suggests "wherein the code generation information specifies a type of graphical program to create in response to the recorded one or more functions; wherein the graphical program is created in accordance with the specified graphical program type", as recited in claim 22.

As discussed above, Morris nowhere discloses automatically creating a graphical program, and so does not, and cannot, teach or suggest that the user can specify code generation information that specifies a type of graphical program to create, wherein this type of graphical program is then automatically created.

The Examiner relied upon Morris col. 3, lines 5-15 to teach the features of claims 22, 46, and 68. Morris teaches and discloses: "Programs of the prior art each have their own object definition standards. That is, for an object to be incorporated into their system, it must meet that program's standards. No prior art program is available which will allow addition of objects not designed to its standard without the necessity of writing additional code to perform the interface to that object" (Morris col. 3, lines 12-18) (*emphasis added*). In other words, Morris teaches and discloses different object types but not different graphical program types as described in Appellant's Specification as summarized above on page 5 of this Appeal Brief. Further, as noted above, Morris does not teach any type of automatic generation of a graphical program, and certainly does not allow a user to specify a type of graphical program that is automatically created.

In the Response to Arguments, the Examiner asserts that Morris inherently discloses automatically generating a graphical program according to a graphical program type specified by the user, stating that "as long as the graphical programs generated have varying formats or layouts, there is represented graphical programs of different types". Appellant respectfully submits that the Examiner has improperly read into Morris Appellant's claimed feature, absent any actual disclosure in Morris describing this

feature. Nowhere does Morris teach or suggest, or even hint at receiving user input specifying code generation information that specifies a type of graphical program to create in response to the recorded one or more functions.

Thus, for at least the reasons presented, Appellant respectfully submits that claim 22 is patentably distinguished over the cited art, and is thus allowable.

Claims 46 and 68 include limitations similar to claim 22, and so the arguments presented above apply with equal force to these claims, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 46 and 68, and any claims respectively dependent therefrom, are patentably distinguished over the cited art, and are allowable.

Claims 24, 48, and 69

Appellant respectfully submits that Morris neither teaches or suggests ". . .wherein said automatically generating the graphical program comprises enabling the graphical program to receive user input during program operation, wherein the user input specifies values for the specified one or more input parameters. . . (*emphasis added*)" as recited in claim 24, at least for the reason that Morris fails to disclose automatic generation of a graphical program. Furthermore, Morris nowhere discloses that such an automatic generation of a graphical program would enable the automatically created graphical program to receive user input during program operation.

Rather, the cited portion of Morris describes an integrated development environment (IDE) which receives user input during creation of a script:

> . . .the user of the system drags an object into the view. The system [IDE] simultaneously generates the underlying script which reflects the object and the properties associated with the object. The user's double click on the icon representing the object presents a menu list of the properties for that object the settings of which can be changed according to the user's requirements. If a separate dialog box is available for a particular property, clicking on the property gets the user to the dialog box. As a property setting is defined, the system records that definition in the associated

> underlying script. In addition, the system user can elect to have all the four views synchronized so that each view is updated for every change made in any view to the underlying script. (Morris col. 6, lines 23-36) (*emphasis added*)

Note that the above text refers to dialog boxes used during program creation, not during program operation, i.e., execution of the program after it is created.

In the Response to Arguments, the Examiner asserts that program operation refers not only to program execution, but processes or actions related to the program, e.g., configuration of the program. Appellant respectfully disagrees, noting the configuring a program is a user action, not program operation, and respectfully submits that one of ordinary skill in the art would readily understand that "program operation" refers to runtime, since the program is only capable of operating when it is executing. For example, when a program is being configured by a user, the program itself is not "operating".

Thus, Appellant respectfully submits that claim 24 is patentably distinguished over both Morris. Accordingly, Appellant respectfully submits that, at least for the reasons presented, claim 24 is allowable.

Claims 48 and 69 include limitations similar to claim 24, and so the arguments presented above apply with equal force to these claims, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 48 and 69 are patentably distinguished over Morris, and are thus allowable.

Claims 25, 49, 70

Appellant respectfully submits that Morris neither teaches nor suggests that ". . . automatically generating the graphical program comprises generating portions of graphical code . . . and linking the portions of graphical code together" as recited in claim 25. Not only does Morris fail to disclose automatically generating a graphical program, but Morris nowhere mentions or even hints at automatic code generation that includes automatically linking portions of graphical code together.

Cited col.1:61-col.2:4 simply describes program development via code modules, and nowhere mentions automatically generating a graphical program by linking graphical code together.

Thus, Appellant respectfully submits that claim 25 is patentably distinguished over Morris, and so, at least the reasons presented, claim 25 is allowable.

Claims 49 and 70 include limitations similar to claim 25, and so the arguments presented above apply with equal force to these claims, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 49 and 70 are patentably distinguished over Morris, and thus allowable.

Claims 26, 27-29, 50 and 51

Appellant respectfully submits that Morris nowhere teaches or suggests ". . .wherein generating each portion of graphical code comprises connecting the node inputs and outputs together in order to implement the function with which the portion of graphical code is associated" as recited in claim 26.

Examiner relied upon Morris col. 5, lines 40-50 to teach this feature, which discloses "The four views are different ways to make and look at the same script. Each has its own special perspective and strength. FIG. 2 shows a Map view of a simple program in which the flow of the program and the flow of data is displayed" (Morris col. 5, lines 40-44) (*emphasis added*). Morris does not teach or disclose a portion of a graphical code; instead Morris teaches and discloses an entire "simple program". Appellant respectfully submits that Morris teaches and discloses a static view in col. 5, lines 40-50 and Morris' Figure 2, which does not teach connecting nodes to form a function. Moreover, as discussed above, Morris fails to disclose automatically generating a graphical program, and so does not, and cannot, teach or suggest automatically generating each portion of graphical code, including connecting the node inputs and outputs together in order to implement the function with which the portion of graphical code is associated, as claimed.

Thus, Appellant respectfully submits that claim 26 is patentably distinguished over Morris, and so, for at least the reasons presented, claim 26 is allowable.

Claim 50 includes limitations similar to claim 26 and so the arguments presented above apply with equal force to claim 50, as well. Appellant respectfully submits that, for at least the reasons presented above, claim 50 is patentably distinguished over Morris, and is thus allowable.

Claims 30 and 52

Appellant respectfully submits that Morris neither teaches nor suggests ". . .wherein generating the portion of graphical code that implements a particular function utilizes the database information retrieved for the particular function (*emphasis added*)" as recited by claim 30.

Examiner relied upon "Mass Storage" of Morris' Figure 1 to teach this feature. Appellant respectfully notes that while a database may comprise various mass storage units, a mass storage unit does not teach or suggest using a database. Further, Morris fails to teach or suggest that database information is used in automatically generating a portion of graphical code.

Thus, Appellant respectfully submits that claim 30 is patentably distinguished over Morris, and so, at least for the reasons presented, claim 30 is allowable.

Claim 52 includes limitations similar to claim 30 and so the arguments presented above apply with equal force to claim 52, as well. Appellant respectfully submits that, for at least the reasons presented above, claim 52 is patentably distinguished over Morris, and is thus allowable.

Claims 6, 31, 34, 35, 58

Appellant respectfully submits that independent claim 31 includes similar limitations as claim 1, but where the one or more functions are stored by a prototyping environment application, and where the automatic generation of the graphical program is performed by prototyping environment. Thus, the arguments presented above with respect to claim 1 regarding automatically generating the graphical program (where the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, and where the graphical program implements the algorithm) without direct user input, apply with equal force to this claim. Moreover,

Appellant notes that Morris, Oka, and Yamada all fail to teach or suggest, or even mention, a prototyping environment, and so the cited references do not, and cannot, teach or suggest this feature of claim 31.

Thus, for at least the reasons provided, Appellant submits that claim 31 is patentably distinct and non-obvious over the cited art, and is thus allowable.

Claims 3, 55, 71, 74, 77, 79, 81, 86, 87, and 88

Appellant respectfully submits that independent claim 71 includes similar limitations as claim 1, but where the automatic generation of the graphical program is performed based on a prototype created in response to user input. Thus, the arguments presented above with respect to claim 1 regarding automatically generating the graphical program (where the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, and where the graphical program implements the algorithm) without direct user input, apply with equal force to this claim. Moreover, Appellant notes that Morris, Oka, and Yamada all fail to teach or suggest, or even mention, a prototype, and so the cited references do not, and cannot, teach or suggest this feature of claim 71.

Thus, for at least the reasons provided, Appellant submits that claim 71 is patentably distinct and non-obvious over the cited art, and is thus allowable.

Independent claim 81 includes limitations similar to claim 71, and so the arguments presented above apply with equal force to this claim, as well. Appellant respectfully submits that, for at least the reasons presented above, claim 81 is patentably distinguished over the cited art, and is thus allowable.

Claim 4, 72, 82

In addition to the features and limitations of claim 1 and 3, discussed above, claim 4 includes the limitation "wherein the prototype comprises a prototype in at least one of the disciplines from the group consisting of: image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, workflow processes, and robotics." Nowhere do the cited references disclose a prototype

directed to any of these disciplines. Thus, for at least these reasons, Appellant respectfully submits that claim 4 is patentably distinguished over the cited art, and thus allowable, as are claims 72 and 82.

Claims 32, 73, 83, 84, 86, and 88

Appellant respectfully submits that claim 32, in addition to the limitations of claim 31, but further includes the limitation that storing the functions includes creating a prototype (from which the graphical program is automatically generated). Thus, the arguments presented above with respect to claims 31 regarding automatically generating the graphical program (where the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, and where the graphical program implements the algorithm) without direct user input, apply with equal force to claim 32, as do the arguments provided above with respect to claim 71. Since none of the cited references teach or suggest or even mention a prototype or a prototyping environment, the cited references do not, and cannot, teach or suggest this feature of claim 32.

Claim 33

In addition to the features and limitations of claim 32, discussed above, claim 33 includes the limitation "wherein the prototyping environment application is a prototyping environment application in at least one of the disciplines from the group consisting of: image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, workflow processes, and robotics" Nowhere do the cited references disclose a prototyping environment directed to any of these disciplines. Thus, for at least these reasons, Appellant respectfully submits that claim 33 is patentably distinguished over the cited art, and thus allowable.

Claims 36 and 37

Appellant respectfully submits that Morris neither teaches nor suggests ". . .wherein the graphical program creation program is executable to automatically generate

the graphical program in response to said prototyping environment application calling the graphical program creation program" as recited by claim 36.

The Examiner relied upon Morris col.5:8-13 and col.7:4-7 to teach this feature.

Appellant respectfully notes that Morris col.5:8-13 states: "FIG. 1 is a block diagram showing the basic functional units of a modern digital computer with which the computer implemented applications development system of the invention works. Information may be stored on magnetic or optical media for access or be available through a connection to a remote information source."

Col. 7:4-7 state: "Programs authored by the computer implemented application development system of this invention run faster under its run time program than to object oriented programs developed and run under Microsoft Visual Basic."

As argued above, Morris nowhere teaches or suggests automatically creating a graphical program, and so Morris does not, and cannot, teach or suggest a graphical program creation program being executable to automatically generate a graphical program in response to a prototyping environment application calling the graphical program creation program. Nor do Oka or Yamada disclose this feature.

Thus, Appellant respectfully submits that claim 36 is patentably distinguished over the cited art, and thus allowable.

Claim 39

In addition to the features and limitations of claim 31, discussed above, claim 39 includes the limitation "wherein said automatically generating the graphical program comprises automatically including and connecting the nodes in the graphical program without direct user input (*emphasis added*)", which is not disclosed by either Morris or Oka (or Yamada), as discussed above with respect to claim 9.

Thus, Appellant respectfully submits that claim 39 is patentably distinguished over the cited art, and so for at least for the reasons presented, claim 39 is allowable.

Claim 41

In addition to the features and limitations of claim 31, discussed above, claim 41 includes the limitation "wherein the graphical program is a graphical data flow program."

Nowhere do any of the cited references even mention a graphical data flow program. Thus, for at least these reasons, Appellant respectfully submits that claim 41 is patentably distinguished over the cited art, and thus allowable.

Claim 56

In addition to the features and limitations of claims 55, discussed above, claim 56 includes the limitation "wherein the prototype comprises a prototype in at least one of the disciplines from the group consisting of: image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, workflow processes, and robotics." Nowhere do the cited references disclose a prototype directed to any of these disciplines. Thus, for at least these reasons, Appellant respectfully submits that claim 4 is patentably distinguished over the cited art, and thus allowable.

Claims 76, 85

In addition to the features and limitations of claims 71 and 81, discussed above, claims 76 and 85 each includes limitations similar to claim 9 (and 39), specifically each includes the feature that ". . .wherein said automatically generating the graphical program comprises programmatically including and connecting the nodes in the graphical program without direct user input", and so the arguments presented above apply with equal force to claims 76 and 85, as well. Appellant respectfully submits that, for at least the reasons presented above, claims 76 and 85 are patentably distinguished over the cited art, and are thus allowable.

Claims 80, 89

Appellant respectfully submits that Morris neither teaches nor suggests that ". . . automatically generating the graphical program comprises generating portions of graphical code . . . and linking the portions of graphical code together" as recited in claim 80. Not only does Morris fail to disclose automatically generating a graphical program, but Morris nowhere mentions or even hints at automatic code generation that includes automatically linking portions of graphical code together.

Cited col.1:61-col.2:4 simply describes program development via code modules, and nowhere mentions automatically generating a graphical program by linking graphical code together.

Thus, Appellant respectfully submits that claim 80 is patentably distinguished over Morris, and so, for at least the reasons presented above, claim 80 is allowable.

Claim 85

In addition to the features and limitations of claim 81, discussed above, claim 85 includes the limitation "wherein said automatically generating the graphical program comprises automatically including and connecting the nodes in the graphical program without direct user input (*emphasis added*)", which is not disclosed by either Morris or Oka (or Yamada), as discussed above with respect to claim 9 (and 39).

Thus, Appellant respectfully submits that claim 85 is patentably distinguished over the cited art, and so for at least for the reasons presented, claim 85 is allowable.

Claims 78, 87

In addition to the features and limitations of claims 71 and 81, discussed above, claim 78 includes the limitation "wherein said creating the prototype in response to user input comprises creating a diagrammatic model of the algorithm" which is not disclosed by either Morris or Oka (or Yamada). For example, none of the cited references mentions a prototype at all, and to the cited art does not, and cannot disclose this feature of claim 78. Thus, Appellant respectfully submits that claim 78 is patentably distinguished over the cited art, and so for at least for the reasons presented, claim 78 is allowable.

Thus, for at least the reasons provided above, Appellant respectfully requests removal of the section 103 rejection of claims 1-16, 21-37, 39-43, 45-59, 61-65, 67-74, and 76-90.

Second Ground of Rejection:

Claims 17-20, 44, and 66 are finally rejected under 35 U.S.C. 103(a) as being unpatentable over Morris, Oka, and Yamada, in further view of Shi et al. (U.S. Patent No. 5,623,659, hereinafter "Shi"). Different groups of claims are addressed under their respective subheadings.

Claims 17-20, 44, and 66

Appellant respectfully submits that independent claims 1, 31, 53, 71, 81, and 90 have been argued above to overcome rejections under 35 U.S.C. 103 as being unpatentable over Morris, Oka, and Yamada. Appellant respectfully submits that, at least, based on the arguments presented above for claims 1, 3, 5-7, 12, 13, 32, 42, 53, 55, 57-59, 64, 73, 77, 83, 86, and 90, independent claims 1, 31, 53, 71, 81, and 90 are nonobvious in further view of Shi. Appellant respectfully notes "If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)" as stated in the MPEP §2143.03. Accordingly, Appellant respectfully submits that claims 17-20, 44, and 66 are patentably distinct and non-obvious, and thus allowable.

Furthermore, Appellant respectfully submits that Morris, Oka, Yamada, and/or Shi nowhere teach or suggest ". . .locking the association between the script and the graphical program, wherein said locking prevents user editing of the graphical program" as recited by claim 17.

The Office Action admits that Morris, Oka, and Yamada fail to teach this features, but asserts that Shi remedies this admitted deficiency, citing col.2:5-11. However, Appellant respectfully notes that the cited text describes locking portions of data sets, and nowhere discloses or even mentions locking the association between a script and a graphical program to prevent or manage user editing of the graphical program.

More generally, Shi teaches user editing of a locked portion of a data set: ". . .the request is granted by allowing write access to the first portion [of the data set]. A write lock is placed on the first portion of the data set to prevent other users from changing it, while allowing other users read access to the first portion" (Shi Abstract) (*emphasis added*). In other words, Shi teaches that a first user has write or user editing access to the

first portion of the data set while other users are prevented from writing to or editing the first portion of the data set.

Moreover, since Shi teaches user editing of a "locked" portion of a data set, Appellant respectfully submits that Shi teaches away from locking an association which prevents user editing. Accordingly, Appellant respectfully submits that a *prima facie* case of obviousness has not been established to reject claim 17. *See In re Haruna*, 249 F.3d 1327, 1335 (Fed. Cir. 2001) (outlining elements of a reference teaching away and rebutting a *prima facie* case of obviousness).

In the Response to Arguments, the Examiner asserts that Shi's locking data sets "can represent various types of data including an association between a script and a graphical program", even though no such association is even mentioned in Shi. The Examiner further asserts that Shi's data set locking method could be used to keep up with versions of one distinct system component to "attain version control"; however, Appellant respectfully submits that this is not germane to Appellant's claimed invention as represented in claim 17 at least for the reasons that Shi's version control is not directed to controlling editing of a graphical program, and that Shi nowhere discloses locking a script that is associated with a graphical program.

Furthermore, Appellant respectfully submits that Shi nowhere teaches or suggests storing an association between a script and a graphical program, noting that Shi is not related at all to graphical programming, and certainly does not teach or suggest locking an association between a script and a graphical program. Thus, Appellant submits that Shi is non-analogous art with respect to Appellant's invention as represented in claim 17.

Further, since Shi does not relate to graphical programming at all, nor to automatically generating programs of any type, Appellant respectfully submits that there is no proper teaching, suggestion or motivation to combine Shi with the other references; nor is such a motivation provided in the references or in the prior art. The only motivation to combine suggested by the Examiner is "to have more control over the user's manipulations of the graphical program", which is simply a statement of presumed benefit of Appellant's claimed invention, and is neither indicated nor suggested by the cited art, which, as noted above, makes no reference to an association between a script and a graphical program, nor locking such an association as claimed. As held by the U.S.

Court of Appeals for the Federal Circuit in *Ecolochem Inc. v. Southern California Edison Co.*, an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis. Thus, Appellant respectfully submits that Morris, Oka, Yamada, and Shi are not available for use in a prima facie case of obviousness.

Furthermore, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular. . .Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination. Appellant respectfully submits that there is no suggestion in the prior art for combining Morris, Oka, Yamada, and/or Shi, and that even were the references properly combinable, the alleged combination would still not produce the features of claim 17.

Thus, Appellant respectfully submits that a *prima facie* case of obviousness has not been established to reject claim 17. Accordingly, Appellant respectfully submits that, at least for the reasons presented, claim 17 and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and thus allowable.

Claims 44 and 66 each includes limitations similar to claim 17, and so the arguments presented above apply with equal force to these claims, as well. Thus, Appellant respectfully submits that for at least the reasons provided above, claims 44 and 66, and those claims respectively dependent therefrom, are patentably distinguished over Morris, Oka, Yamada, and Shi, taken both singly and in combination, and so are allowable.

Removal of the section 103 rejection of claims 17-20, 44, and 66 is respectfully requested.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-7, 9-37, 39-59, 61-74, and 76-90 was erroneous, and reversal of Examiner's decision is respectfully requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-44300/JCH.

Respectfully submitted,

/Jeffrey C. Hood/

Jeffrey C. Hood, Reg. #35198
Attorney for Appellant(s)

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8800

Date: September 13, 2007 JCH/MSW

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A method of creating a graphical program to perform an algorithm, the method comprising:

recording one or more functions in response to user input, wherein the one or more functions specify the algorithm; and

automatically generating the graphical program in response to the recorded one or more functions, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm;

wherein said automatically generating the graphical program comprises automatically including the nodes in the graphical program, wherein said automatically including the nodes in the graphical program is performed without direct user input selecting the nodes.

2. The method of claim 1, further comprising:

performing the one or more functions in response to user input;

wherein said recording the one or more functions is performed in response to said performing the one or more functions.

3. The method of claim 1,

wherein said recording the one or more functions comprises creating a prototype.

4. The method of claim 3,

wherein the prototype comprises a prototype in at least one of the disciplines from the group consisting of:

image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, workflow processes, and robotics.

5. The method of claim 1,

wherein said recording the one or more functions is performed in response to user input received via a graphical user interface (GUI).

6. The method of claim 5,

wherein the graphical user interface is associated with a prototyping environment application.

7. The method of claim 5,

wherein the user input comprises selecting the functions from one or more of a menu or palette.

9. The method of claim 1,

wherein said automatically generating the graphical program comprises automatically including and connecting the nodes in the graphical program without direct user input.

10. The method of claim 1, further comprising:

executing the graphical program to perform the algorithm.

11. The method of claim 1,

wherein the graphical program includes a block diagram portion and a user interface panel portion.

12. The method of claim 1,

wherein the graphical program is a graphical data flow program.

13. The method of claim 1,

wherein said automatically generating the graphical program comprises automatically including one or more nodes corresponding to respective ones of the one or more functions in the graphical program.

14. The method of claim 1, wherein the recorded one or more functions comprise a script, the method further comprising:

creating an association between the script and the graphical program;

modifying the script to create a new script in response to user input after said creating the association; and

modifying the graphical program according to the new script to create a new graphical program.

15. The method of claim 14,

wherein said modifying the graphical program according to the new script uses the association between the script and the graphical program;

wherein the association remains between the new script and the new graphical program.

16. The method of claim 14, further comprising:

receiving user input indicating a desire to change the graphical program;

displaying script information of the script;

modifying the script information in response to user input; and

modifying the graphical program after said modifying the script information.

17. The method of claim 14, further comprising:

creating an association between the script and the graphical program;

locking the association between the script and the graphical program, wherein said locking prevents user editing of the graphical program.

18. The method of claim 17, further comprising:

unlocking the association between the script and the graphical program in response to user input after said locking;

directly changing the graphical program in response to user input after said unlocking.

19. The method of claim 18,

wherein said unlocking removes the association between the script and the graphical program.

20. The method of claim 17, further comprising:

modifying the graphical program in response to user input after said generating the graphical program and after said creating the association between the script and the graphical program;

determining if an association exists between the script and the graphical program in response to said modifying the graphical program; and

removing the association between the script and the graphical program in response to said modifying.

21. The method of claim 1, further comprising:

receiving user input specifying code generation information;

wherein said automatically generating the graphical program utilizes the code generation information.

22. The method of claim 21,

wherein the code generation information specifies a type of graphical program to create in response to the recorded one or more functions;

wherein the graphical program is created in accordance with the specified graphical program type.

23. The method of claim 22,

wherein the graphical program type specifies a particular graphical programming environment;

wherein the graphical program is created in a file format that is usable by the particular graphical programming environment.

24. The method of claim 21,

wherein a plurality of parameters are associated with the one or more functions, wherein each parameter is an input parameter which provides input to a function or an output parameter which accepts output from a function;

wherein the code generation information specifies one or more of the input parameters which are desired to be interactively changeable or one or more of the output parameters which are desired to be interactively viewable;

wherein said automatically generating the graphical program comprises enabling the graphical program to receive user input during program operation, wherein the user input specifies values for the specified one or more input parameters;

wherein said automatically generating the graphical program comprises enabling the graphical program to display output during program operation, wherein the output indicates values for the specified one or more output parameters.

25. The method of claim 1,

wherein said automatically generating the graphical program comprises:

generating portions of graphical code, wherein each portion of graphical code implements one of the functions;

linking the portions of graphical code together.

26. The method of claim 25,

wherein each portion of graphical code includes one or more graphical program nodes, wherein each node has one or more inputs or outputs;

wherein generating each portion of graphical code comprises connecting the node inputs and outputs together in order to implement the function with which the portion of graphical code is associated.

27. The method of claim 26,

wherein linking a first portion of graphical code to a second portion of graphical code comprises connecting an output of a node in the first portion of graphical code to an input of a node in the second portion of graphical code.

28. The method of claim 26,

wherein at least one of the functions has an associated input parameter;

wherein each portion of code that implements a function that has an associated input parameter includes a node that has an input for receiving a value for the input parameter;

wherein each portion of code that implements a function that has an associated input parameter includes a leaf node that has an output for providing a value for the input parameter;

wherein the leaf node output for providing the parameter value is connected to the node input for receiving the parameter value.

29. The method of claim 26,

wherein at least one of the functions has an associated output parameter;

wherein each portion of code that implements a function that has an associated output parameter includes a node that has an output for providing a value for the output parameter;

wherein each portion of code that implements a function that has an associated output parameter includes a leaf node that has an input for receiving a value for the output parameter;

wherein the leaf node input for receiving the parameter value is connected to the node output for providing the parameter value.

30. The method of claim 25, further comprising:

for each function, retrieving information associated with the function from a database;

wherein generating the portion of graphical code that implements a particular function utilizes the database information retrieved for the particular function.

31. A system for creating a graphical program to perform an algorithm, the system comprising:

a processor;

a memory coupled to the processor which stores a prototyping environment application;

a user input device which receives user input;

wherein the prototyping environment application is executable in response to the user input to store one or more functions in the memory, wherein the one or more functions specify the algorithm;

wherein the prototyping environment application is executable to automatically generate a graphical program in response to the stored one or more functions, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm specified by the one or more functions;

wherein, in automatically generating the graphical program, the prototyping environment application is executable to automatically include the nodes in the graphical program without direct user input selecting the nodes.

32. The system of claim 31,

wherein said storing the one or more functions in the memory comprises creating a prototype.

33. The system of claim 32,

wherein the prototyping environment application is a prototyping environment application in at least one of the disciplines from the group consisting of:

image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, workflow processes, and robotics.

34. The system of claim 31,

wherein the prototyping environment application includes a graphical user interface (GUI);

wherein said storing the one or more functions in the memory is performed in response to user input received via the graphical user interface.

35. The system of claim 34,

wherein the user input comprises selecting the functions from one or more of a menu or palette.

36. The system of claim 31, further comprising:

a graphical program creation program stored in the memory;

wherein the prototyping environment application is executable to call the graphical program creation program;

wherein the graphical program creation program is executable to automatically generate the graphical program in response to said prototyping environment application calling the graphical program creation program.

37. The system of claim 36,

wherein the graphical program creation program is a graphical programming development environment application.

39. The system of claim 31,

wherein said automatically generating the graphical program comprises automatically including and connecting the nodes in the graphical program without direct user input.

40. The system of claim 31,

wherein the graphical program includes a block diagram portion and a user interface portion.

41. The system of claim 31,

wherein the graphical program is a graphical data flow program.

42. The system of claim 31,

wherein said automatically generating the graphical program comprises automatically including one or more nodes corresponding to respective ones of the one or more functions in the graphical program.

43. The system of claim 36,

wherein the stored one or more functions comprise a script;

wherein the memory stores an association between the script and the graphical program;

wherein the prototyping environment application is executable to utilize the association to modify the script to create a new script in response to user input;

wherein the graphical program creation program is executable to modify the graphical program according to the new script to create a new graphical program.

44. The system of claim 43,

wherein the memory stores an association between the script and the graphical program;

wherein the memory stores information specifying that the association between the script and the graphical program is locked, wherein said locking prevents user editing of the graphical program.

45. The system of claim 31,

wherein the prototyping environment application is executable to receive user input specifying code generation information, wherein the code generation information specifies information to use in generating the graphical program.

46. The system of claim 45,

wherein the code generation information specifies a type of graphical program to create in response to the stored one or more functions;

wherein the graphical program is created in accordance with the specified graphical program type.

47. The method of claim 46,

wherein the graphical program type specifies a particular graphical programming environment;

wherein the graphical program is created in a file format that is usable by the particular graphical programming environment.

48. The system of claim 45,

wherein a plurality of parameters are associated with the functions, wherein each parameter is an input parameter which provides input to a function or an output parameter which accepts output from a function;

wherein the code generation information specifies one or more of the input parameters which are desired to be interactively changeable or one or more of the output parameters which are desired to be interactively viewable;

wherein said automatically generating the graphical program comprises enabling the graphical program to receive user input during program operation, wherein the user input specifies values for the specified one or more input parameters;

wherein said automatically generating the graphical program comprises enabling the graphical program to display output during program operation, wherein the output indicates values for the specified one or more output parameters.

49. The system of claim 31,

wherein said automatically generating the graphical program comprises:

generating portions of graphical code, wherein each portion of graphical code implements one of the functions;

linking the portions of graphical code together.

50. The system of claim 49,

wherein each portion of graphical code includes one or more graphical program nodes, wherein each node has one or more inputs or outputs;

wherein generating each portion of graphical code comprises connecting the node inputs and outputs together in order to implement the function with which the portion of graphical code is associated.

51. The system of claim 50,

wherein linking a first portion of graphical code to a second portion of graphical code comprises connecting an output of a node in the first portion of graphical code to an input of a node in the second portion of graphical code.

52. The system of claim 49,

wherein, for each function, information associated with the function is retrieved from a database;

wherein generating the portion of graphical code that implements a particular function utilizes the database information retrieved for the particular function.

53. A memory medium comprising program instructions executable to:

record one or more functions in response to user input, wherein the one or more functions specify an algorithm; and

automatically generate a graphical program in response to the recorded one or more functions, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm;

wherein said automatically generating the graphical program comprises automatically generating graphical code in the graphical program without direct user input.

54. The memory medium of claim 53, further comprising program instructions executable to:

perform the one or more functions in response to user input;

wherein said recording the one or more functions is performed in response to said performing the one or more functions.

55. The memory medium of claim 53,

wherein said recording the one or more functions comprises creating a prototype.

56. The memory medium of claim 55,

wherein the prototype comprises a prototype in at least one of the disciplines from the group consisting of:

image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, workflow processes, and robotics.

57. The memory medium of claim 53,

wherein said recording the one or more functions is performed in response to user input received via a graphical user interface (GUI).

58. The memory medium of claim 57,

wherein the graphical user interface is associated with a prototyping environment application.

59. The memory medium of claim 57,

wherein the user input comprises selecting the functions from one or more of a menu or palette.

61. The memory medium of claim 53,

wherein said automatically generating the graphical program comprises automatically including and connecting the nodes in the graphical program without direct user input.

62. The memory medium of claim 53, further comprising program instructions executable to:

execute the graphical program to perform the algorithm.

63. The memory medium of claim 53,

wherein the graphical program includes a block diagram portion and a user interface panel portion.

64. The memory medium of claim 53,

wherein said automatically generating the graphical program comprises automatically including one or more nodes corresponding to respective ones of the one or more functions in the graphical program.

65. The memory medium of claim 53, wherein the recorded one or more functions comprise a script, the memory medium further comprising program instructions executable to:

create an association between the script and the graphical program;

modify the script to create a new script in response to user input after said creating the association; and

modify the graphical program according to the new script to create a new graphical program.

66. The memory medium of claim 65, further comprising program instructions executable to:

create an association between the script and the graphical program;

lock the association between the script and the graphical program, wherein said locking prevents user editing of the graphical program.

67. The memory medium of claim 53, further comprising program instructions executable to:

receive user input specifying code generation information;

wherein said automatically generating the graphical program utilizes the code generation information.

68. The memory medium of claim 67,

wherein the code generation information specifies a type of graphical program to create in response to the recorded one or more functions;

wherein the graphical program is created in accordance with the specified graphical program type.

69. The memory medium of claim 67,

wherein a plurality of parameters are associated with the one or more functions, wherein each parameter is an input parameter which provides input to a function or an output parameter which accepts output from a function;

wherein the code generation information specifies one or more of the input parameters which are desired to be interactively changeable or one or more of the output parameters which are desired to be interactively viewable;

wherein said automatically generating the graphical program comprises enabling the graphical program to receive user input during program operation, wherein the user input specifies values for the specified one or more input parameters;

wherein said automatically generating the graphical program comprises enabling the graphical program to display output during program operation, wherein the output indicates values for the specified one or more output parameters.

70. The memory medium of claim 53,

wherein said automatically generating the graphical program comprises:

generating portions of graphical code, wherein each portion of graphical code implements one of the functions;

linking the portions of graphical code together.

71. A method of creating a graphical program to perform an algorithm, the method comprising:

creating a prototype in response to user input, wherein the prototype specifies the algorithm; and

automatically generating the graphical program in response to the prototype, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm;

wherein said automatically generating the graphical program comprises automatically including the nodes in the graphical program, wherein said automatically including the nodes in the graphical program is performed without direct user input selecting the nodes.

72. The method of claim 71,

wherein the prototype comprises a prototype in at least one of the disciplines from the group consisting of:

image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, telecommunications, workflow processes, and robotics.

73. The method of claim 71,

wherein the user input is received via a graphical user interface (GUI) associated with a prototyping environment application.

74. The method of claim 73,

wherein the user input comprises selecting one or more functions from one or more of a menu or palette.

76. The method of claim 71,

wherein said automatically generating the graphical program comprises programmatically including and connecting the nodes in the graphical program without direct user input.

77. The method of claim 71,

wherein said automatically generating the graphical program comprises automatically including one or more function nodes in the graphical program.

78. The method of claim 71,

wherein said creating the prototype in response to user input comprises creating a diagrammatic model of the algorithm.

79. The method of claim 71,

wherein said creating the prototype in response to user input comprises recording one or more functions in response to user input;

wherein the recorded one or more functions specify the algorithm.

80. The method of claim 79,

wherein said automatically generating the graphical program comprises:

generating portions of graphical code, wherein each portion of graphical code implements one of the functions;

linking the portions of graphical code together.

81. A memory medium comprising program instructions for creating a graphical program to perform an algorithm, wherein the program instructions are executable to implement:

creating a prototype in response to user input, wherein the prototype specifies the algorithm; and

automatically generating the graphical program in response to the prototype, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm;

wherein said automatically generating the graphical program comprises automatically generating graphical code in the graphical program without direct user input.

82. The memory medium of claim 81,

wherein the prototype comprises a prototype in at least one of the disciplines from the group consisting of:

image processing, machine vision, image analysis, process control, industrial automation, test and measurement, simulation, telecommunications, workflow processes, and robotics.

83. The memory medium of claim 81,

wherein the user input is received via a graphical user interface (GUI) associated with a prototyping environment application.

84. The memory medium of claim 83,

wherein the user input comprises selecting one or more functions from one or more of a menu and a palette.

85. The memory medium of claim 81,

wherein said automatically generating the graphical program comprises programmatically including and connecting the nodes in the graphical program without direct user input.

86. The memory medium of claim 81,

wherein said automatically generating the graphical program comprises automatically including one or more function nodes in the graphical program.

87. The memory medium of claim 81,

wherein said creating the prototype in response to user input comprises creating a diagrammatic model of the algorithm.

88. The memory medium of claim 81,

wherein said creating the prototype in response to user input comprises recording one or more functions in response to user input;

wherein the recorded one or more functions specify the algorithm.

89. The memory medium of claim 88,

wherein said automatically generating the graphical program comprises:

generating portions of graphical code, wherein each portion of graphical code implements one of the functions;

linking the portions of graphical code together.

90. A memory medium comprising program instructions executable to:

record one or more functions in response to user input, wherein the one or more functions specify an algorithm; and

automatically generate a graphical program in response to the recorded one or more functions, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program implements the algorithm;

wherein, in automatically generating the graphical program, the program instructions are executable to automatically generate graphical code in the graphical program without direct user input.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.